

Machine Learning 1

Lecture 7.4 - Supervised Learning
Classification - Logistic Regression: Newton-Raphson Optimization

Erik Bekkers

(Bishop 4.3.3)



Logistic Regression for Two Classes

- Given: Dataset $\mathbf{X} = (\mathbf{x}_1, \dots, \mathbf{x}_N)^T$ with binary targets $\mathbf{t} = (t_1, \dots, t_N)^T$ with $t_n \in \{\mathcal{C}_1, \mathcal{C}_2\} = \{1, 0\}$
- Conditional likelihood function:

$$p(\mathbf{t}|\mathbf{X}, \mathbf{w}) = \prod_{n=1}^N p(t_n|\mathbf{x}_n, \mathbf{w}) = \prod_{n=1}^N y_n^{t_n} (1 - y_n)^{1-t_n}$$

$$y_n = p(C_1|\phi_n) = \sigma(w^T \phi_n) \quad \phi_n = \phi(\mathbf{x}_n)$$

- Maximizing the conditional likelihood/minimizing the cross-entropy

$$E(\mathbf{w}) = -\ln p(\mathbf{t}, \mathbf{X}, \mathbf{w}) = -\sum_{n=1}^N t_n \ln y_n + (1 - t_n) \ln(1 - y_n)$$

- $E(\mathbf{w})$: convex, but no closed form solution!

$y_n = \sigma(\mathbf{w}^T \phi_n)$ is nonlinear in \mathbf{w}

Newton-Raphson Iterative Optimization

- Goal: minimize

$$E(\mathbf{w}) = - \sum_{n=1}^N t_n \ln y_n + (1 - t_n) \ln(1 - y_n)$$

- Newton-Raphson iterative optimization scheme:

- Initial guess $\mathbf{w}^{(0)}$

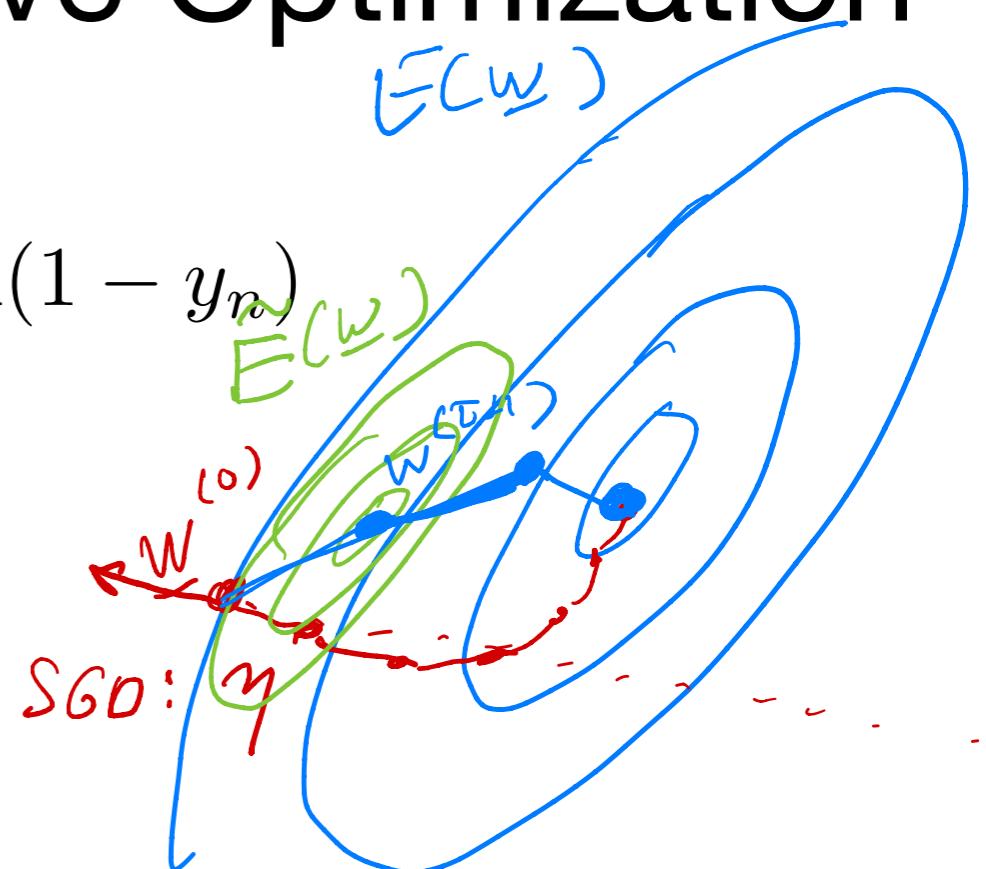
- For $\tau = 1, \dots :$

- I. Approximate $E(\mathbf{w})$ with a quadratic function $\tilde{E}(\mathbf{w})$ around $\mathbf{w}^{(\tau-1)}$

- II. Construct $\mathbf{w}^{(\tau)}$ such that it minimizes $\tilde{E}(\mathbf{w})$

- III. Stop when $\| \mathbf{w}^{(\tau-1)} - \mathbf{w}^{(\tau)} \| = 0$

3. You have found \mathbf{w}^* such that $\frac{\partial}{\partial \mathbf{w}} E(\mathbf{w}^*) = 0$



Newton-Raphson Iterative Optimization

- Given your old estimate $\mathbf{w}^{(\tau-1)}$, approximate $E(\mathbf{w})$ with a second order Taylor expansion around $\mathbf{w}^{(\tau-1)}$

$$E(\mathbf{w}) \approx \tilde{E}(\mathbf{w}^{(n-1)} + \Delta\mathbf{w}) = E(\mathbf{w}^{(n-1)}) + (\Delta\mathbf{w})^T \nabla^T E(\mathbf{w}^{(n-1)}) + \frac{1}{2} (\Delta\mathbf{w})^T \mathbf{H} \Delta\mathbf{w}$$

- Gradient $\nabla E(\mathbf{w}) = \left(\frac{\partial E(\mathbf{w})}{\partial w_0}, \dots, \frac{\partial E(\mathbf{w})}{\partial w_{M-1}} \right)$
- Hessian Matrix: $H_{ij} = \frac{\partial^2 E(\mathbf{w})}{\partial w_i \partial w_j}$ is symmetric
- Choose $\Delta\mathbf{w}$ such that next estimate $\mathbf{w}^{(\tau)} = \mathbf{w}^{(\tau-1)} + \Delta\mathbf{w}$ minimizes $\tilde{E}(\mathbf{w})$

$$\frac{\partial}{\partial \Delta\mathbf{w}} \tilde{E}(\mathbf{w}^{(\tau-1)} + \Delta\mathbf{w}) = \nabla \tilde{E} + (\Delta\mathbf{w})^T \mathbf{H} = 0 \rightarrow \mathbf{H} \Delta\mathbf{w} = -\nabla^T \tilde{E}$$
$$\rightarrow \Delta\mathbf{w} = -\mathbf{H}^{-1} \nabla^T \tilde{E}$$

- Update rule: $\mathbf{w}^{(\tau)} = \mathbf{w}^{(\tau-1)} - \mathbf{H}^{-1} \nabla^T E(\mathbf{w}^{(\tau-1)})$

Newton-Raphson Iterative Optimization

- Update rule: $\mathbf{w}^{(\tau)} = \mathbf{w}^{(\tau-1)} - \mathbf{H}^{-1} \nabla E(\mathbf{w}^{(\tau-1)})$ ($E(\underline{\omega}) = \sum_{n=1}^N E_n(\underline{\omega})$)

- Gradient $\nabla E_n(\mathbf{w})^T = \left(\frac{\partial E_n(\mathbf{w})}{\partial w_0}, \dots, \frac{\partial E_n(\mathbf{w})}{\partial w_{M-1}} \right)^T = (y_n - t_n) \phi_n$

$\Phi \in \mathbb{R}^{N \times M}$

$$\nabla E(\underline{\omega})^T = \sum_{n=1}^N (y_n - t_n) \phi_n = \Phi^T (\underline{y} - \underline{t}) \in \mathbb{R}^M$$

Hessian

$$\mathbf{H}_{ij} = \frac{\partial E(\mathbf{w}^{(\tau-1)})}{\partial w_i \partial w_j} = \frac{\partial}{\partial w_i} \sum_{n=1}^N (y_n - t_n) \phi_j(\mathbf{x}_n) = \sum_{n=1}^N \phi_j(\mathbf{x}_n) \frac{\partial y_n}{\partial w_i}$$

$$= \sum_{n=1}^N y_n (1-y_n) \phi_i \phi_j$$

$$\mathbf{H} = \sum_{n=1}^N y_n (1-y_n) \phi_n \phi_n^T = \Phi^T R \Phi$$

$$R_{nn} = y_n (1-y_n)$$

$$R_{nm} = 0 \text{ if } n \neq m$$

The cross entropy loss is convex

- The error function $E(\mathbf{w})$ is convex when its Hessian is positive definite, meaning $\forall \mathbf{w} \neq \mathbf{0} \in \mathbb{R}^M : \mathbf{w}^T \mathbf{H} \mathbf{w} > 0$

- The Hessian of $E(\mathbf{w})$ is given by $\mathbf{H} = \Phi^T \mathbf{R} \Phi$, with $\mathbf{R} = \text{diag}_{N \times N} \{y_n(1 - y_n)\}$

$$\begin{aligned}\underline{\mathbf{w}}^T \underline{\mathbf{H}} \underline{\mathbf{w}} &= \underline{\mathbf{w}}^T \underline{\Phi}^T R \underline{\Phi} \underline{\mathbf{w}} \\ &= (\underline{\mathbf{w}}^T \underline{\Phi}^T R^{\frac{1}{2}}) R^{\frac{1}{2}} \underline{\Phi} \underline{\mathbf{w}} \\ &= (R^{\frac{1}{2}} \underline{\Phi} \underline{\mathbf{w}})^T (R^{\frac{1}{2}} \underline{\Phi} \underline{\mathbf{w}}) \\ &= \|R^{\frac{1}{2}} \underline{\Phi} \underline{\mathbf{w}}\|^2 > 0\end{aligned}$$

$$\mathbf{R}^{N \times N} \quad \mathbf{R}^{N \times M}$$

$$0 < y_n < 1$$

$$R = R^{\frac{1}{2}} R^{\frac{1}{2}}$$

$$R^{\frac{1}{2}} = \text{diag}(\sqrt{y_n(1-y_n)})$$

Iterative Reweighted Least Squares

- Update rule: $\mathbf{w}^{(\tau)} = \mathbf{w}^{(\tau-1)} - \mathbf{H}^{-1} \nabla E(\mathbf{w}^{(\tau-1)})$
- Gradient: $\nabla E(\mathbf{w}) = \Phi^T (\mathbf{y} - \mathbf{t})$
- Hessian: $\mathbf{H} = \Phi^T \mathbf{R} \Phi$ with $R_{nn} = y_n(1 - y_n)$
- Newton-Raphson update rule for two-class logistic regression:

$$\begin{aligned}\mathbf{w}^{(\tau)} &= \mathbf{w}^{(\tau-1)} - (\Phi^T \mathbf{R} \Phi)^{-1} \Phi^T (\mathbf{y} - \mathbf{t}) \\ &= (\Phi^T R \Phi)^{-1} \{ \Phi^T R \Phi w^{(\tau-1)} - \Phi^T (\mathbf{y} - \mathbf{t}) \} \\ &= (\Phi^T R \Phi)^{-1} \{ \underbrace{\Phi^T R z}_{\text{z}} + \underbrace{R^{-1}(\mathbf{y} - \mathbf{t})}_{\text{R}^{-1}(\mathbf{y} - \mathbf{t})} - \Phi^T (\mathbf{y} - \mathbf{t}) \} \\ &= (\Phi^T R \Phi)^{-1} \Phi^T R z \quad \mathbf{z} = \Phi \mathbf{w}^{(\tau-1)} - \mathbf{R}^{-1}(\mathbf{y} - \mathbf{t})\end{aligned}$$

- Note similarity with ML solution to linear regression:
 $\mathbf{w}_{ML} = (\Phi^T \Phi)^{-1} \Phi^T \mathbf{t}$
- solution to*
argmin $\sum_w r_i (\underline{w}^T \underline{\phi}_i - z_i)^2$

SGD vs Newton-Raphson

NR: + no step size
+ faster convergence
- Hessian
- need H^{-1}

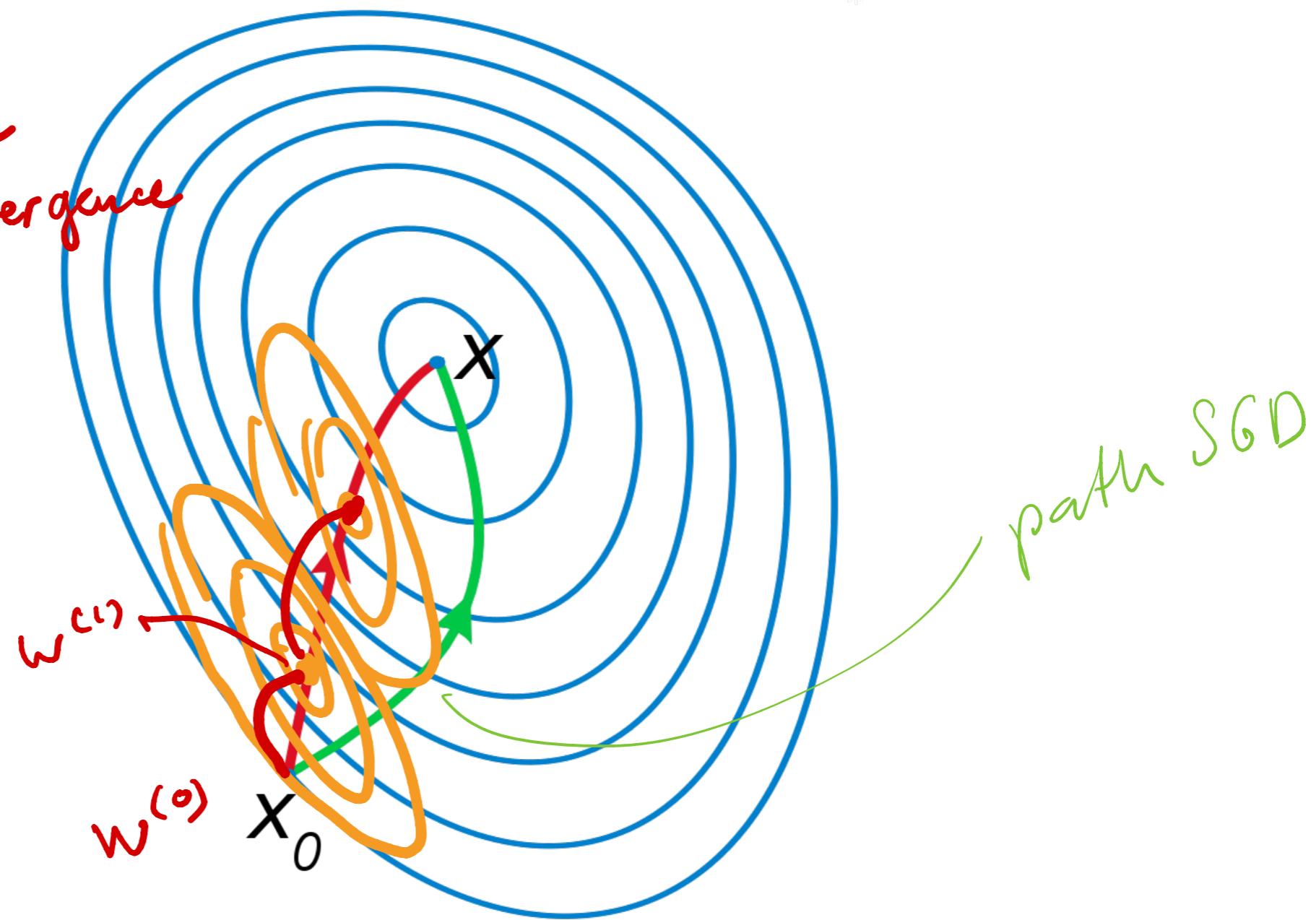


Figure: Green: gradient descent, which always goes in the direction of steepest descent. Red: Newton-Raphson's procedure, which takes into account curvature to take a more direct path. (Wikipedia)